Udacity Data Science Project 2: Data Wrangling Open Street Maps

Project submission by Tony Malerich as partial completion of the Udacity Data Science Nanodegree program

Overview

The second series of the Udacity Data Science program involves data wrangling, and the culmination of the data wrangling course is a project to download, clean, shape, and run a basic analysis of an Open Street Map dataset. This document reports on our efforts with the data wrangling project.

Section 1. Map Selection and Download

I chose to download the OSM data for Albuquerque, New Mexico. Albuquerque is an interesting city situated in the foothills of the Sandia Mountains in the high desert of New Mexico, and gained fame as the site of the AMC smash television series ‘Breaking Bad’. I lived in this city in the land of enchantment for a number of years, and so am familiar with and favor the location, and this helps to pick up on any peculiarities we could encounter in the dataset.

We downloaded the zip file from Map Zen, which expanded to 56.9 MB when uncompressed.

Section 2. Processing the Dataset and Problems Encountered

We began processing the dataset by following the techniques we used in the Lesson 6 Case Study of the Data Wrangling with MongoDB course. In that case study, there were five distinct python functions used to analyze and process the dataset: these functions checked the type and number of tags, audited and standardized various attributes of the tags, and transformed the data to a desirable shape.

For our processing of the Albuquerque map data, we combined the functionality of the case study functions into one python module `map_cleaner.py`. For our use here, we passed the module the path to the Albuquerque OSM file, but also attempted to make the module generic so that we could use it to process any OSM dataset simply by changing the path. As we combined the functions from the case study, the first thing we observed was that each function iterated over all the elements in the dataset; this was ok when we wanted to break things up and run each function separately over a small example dataset, but is certainly inefficient against our 50 MB dataset. So we re-factored the code to iterate once over the elements in the dataset and call the various checks, audits, and shaping from a single loop.

An interesting problem we encountered in cleaning the Albuquerque dataset happened in the audit and standardizing of the street type data. Here, we want to standardize addresses to use only the form "Street" for "st" or "st.", and "Avenue" for "Ave" and so on. First of all, we found an expanded number of street types in this dataset over the example, such as "Boulevard" and "Drive" – not surprising, since this is a much larger dataset than used in the case study. However, the audit took on a new wrinkle over the case study in this aspect: many streets in Albuquerque end with a section designation, such as "Northeast" or "Southwest". The section designation is sometimes given as "Northeast", sometimes as "NE", sometimes as "ne", and should be standardized. Moreover, it breaks the processing we used in the case study, since there we looked at the last string of the address for "st", and here we could have an address such as "1008 Richmond Dr SE", where we need to fix both "Dr" to "Drive" and "SE" to "Southeast". Here are a couple of examples of some of the address names we had to clean, along with the clean version:

```
Mullhacen Pl => Mullhacen Place

Uptown Loop Rd NE => Uptown Loop Road Northeast

LOMAS BLVD NE => LOMAS Boulevard Northeast

Central Ave NE => Central Avenue Northeast

Valley View Dr NW => Valley View Drive Northwest

Menaul Blvd. NE. => Menaul Boulevard Northeast

Comanche Road Norhteast => Comanche Road Northeast

3400 Crest Ave. SE => 3400 Crest Avenue Southeast

16th Street SouthWest => 16th Street Southwest
```

Section 3. Overview of the Map Data

We want to show some overall statistics about the Albuquerque map data.

First we recall that the size of the Albuquerque_new-mexico.osm file is 56.9 MB uncompressed. Running our `map_cleaner` module creates a json database, and the size of the json file is 63.1 MB

The `map_cleaner` module outputs some information regarding the dataset along the way as it cleans and formats the data. Here is some of the output from that module:

```
NUMBER OF ENTRIES WRITTEN TO JSON:

270849

TAGS PRESENT IN THE MAP DATA:

{'bounds': 1,

 'member': 1479,

 'nd': 293381,

 'node': 234148,

 'osm': 1,

 'relation': 138,

 'tag': 210755,

 'way': 36701}



RESULT FROM TAG KEY AUDIT:

{'lower': 93431, 'lower_colon': 115088, 'other': 2229,
'problemchars': 7}



NUMBER OF CONTRIBUTORS TO THE MAP DATA:
```

HERE ARE THE FIRST AND LAST FORMATTED ENTRIES:

```
{'created': {'changeset': '13889129',
             'timestamp': '2012-11-15T23:35:56Z',
             'uid': '26299',
             'user': 'uboot',
             'version': '3'},
 'id': '140740032',
 'pos': [35.1914041, -106.6629759],
 'type': 'node'}
{'created': {'changeset': '29942703',
             'timestamp': '2015-04-03T03:41:04Z',
             'uid': '103107',
             'user': 'EdHillsman',
             'version': '1'},
 'id': '336175903',
 'landuse': 'construction',
 'node_refs': ['3432525863',
               '3432525864',
               '3432525862',
               '3432525869',
               '2062548604',
               '3432525866',
               '3432525865',
               '3432525867',
               '3432525860',
               '3432525856',
               '3432525853',
               '3432525854',
               '3432525863'],
 'pos': [0.0, 0.0],
 'type': 'way'}
```

After we have created the clean json file, we insert it to MongoDB and run some queries with our `query_abq_map` module. Here is output from `query_abq_map` module, along with some of the MongoDB queries we use.

We start by asking MongoDB how many entries we have.

```
num_docs = db.abqMap.find().count()

Number of documents: 270849
```

Next we ask for the number of nodes, and the number of ways.

```
num_nodes = db.abqMap.find({"type":"node"}).count()

Number of nodes: 234146

num_ways = db.abqMap.find({"type":"way"}).count()

Number of ways: 36695
```

Now let us see what are the top amenities in Albuquerque.

```
pipeline = [
     { "$match": {"amenity":{"$exists":1}}},
     { "$group": {"_id":"$amenity", "count":{"$sum":1}}},
     { "$sort" : { "count" : -1 } },
     { "$limit": 10}
]
result = db.abqMap.aggregate(pipeline)

Top amenities:

[{u'_id': u'parking', u'count': 1416},

 {u'_id': u'school', u'count': 259},

 {u'_id': u'place_of_worship', u'count': 228},

 {u'_id': u'restaurant', u'count': 222},

 {u'_id': u'bicycle_parking', u'count': 221},

 {u'_id': u'fast_food', u'count': 174},

 {u'_id': u'fuel', u'count': 113},

 {u'_id': u'cafe', u'count': 56},

 {u'_id': u'post_box', u'count': 53},

 {u'_id': u'bank', u'count': 46}]
```

Here we query for the most popular types of cuisine.

```
pipeline = [
     { "$match": {"amenity": "restaurant" } },
     { "$group": {"_id":"$cuisine", "count" : {"$sum" : 1}
} },
     { "$sort" : { "count" : -1 } },
     { "$limit": 10}
]
result = db.abqMap.aggregate(pipeline)

Top cuisines:

[{u'_id': None, u'count': 132},

 {u'_id': u'mexican', u'count': 16},
 {u'_id': u'pizza', u'count': 11},

 {u'_id': u'regional', u'count': 6},

 {u'_id': u'american', u'count': 6},

 {u'_id': u'burger', u'count': 4},

 {u'_id': u'thai', u'count': 3},

 {u'_id': u'chinese', u'count': 3},

 {u'_id': u'vietnamese', u'count': 3},

 {u'_id': u'asian', u'count': 3}]
```

Section 4. Other Ideas about the Dataset

So some of the queries that we ran against the database show results that are worthy of comment.

First, we find it interesting, and certainly unexpected, that the top amenity listed in Albuquerque is 'parking', with 1416 instances. In all the time I spent living in ABQ, I don't remember dwelling or focusing on parking; in a city like San Francisco, where you can spend half an hour driving around trying to park, I can see the need to identify parking. But in Albuquerque we just drove to where we were going then parked without giving it much thought. How so many listings for parking are logged in the osm data would be interesting to know. Maybe many cities in osm list parking? Observe that 'bicycle_parking' is the 5th most counted amenity. Maybe someone in the ABQ area is really obsessed with parking and has surveyed all the possible locations.

Moving on to the cuisine of Albuquerque, we note that the top count-getter is 'None'. This says that the logging of the restaurant data is incomplete more often than not, so we have identified a deficiency in the data. Another possible deficiency in the data is the fact that we have cuisines listed as 'thai', 'chinese', 'vietnamese', and 'asian'. In that case, what does 'asian' mean? So potentially it would be more convenient to have categories such as 'asian' or 'american', and then more specific sub-categories such as 'vietnamese' or 'burger'. Also it's interesting that New Mexican cuisine doesn't even break the top 10, as I think every other restaurant in Albuquerque is a place serving up hot and spicy green and red chile served over enchiladas or burritos, and that is the food that I crave the most on any visit back to the city. It looks like the Open Street Map community has more work to do in New Mexico.

In so far as uses of the osm data, with all the information that's here, the uses are limited only by our imagination. We could use it to build a web application that identifies cycling routes throughout the city; or to pick out the best places to dine based on neighborhood and type of food.

References:

https://docs.google.com/document/d/1F0Vs14oNEs2idFJR3C_OPxwS6L0HPliOii-QpbmrMo4/pub
http://docs.mongodb.org/manual/reference/operator/aggregation/project/#pipe._S_project
https://www.mongodb.com/reference
http://docs.mongodb.org/manual/indexes/
https://wiki.openstreetmap.org/wiki/OSM_XML
https://mapzen.com/metro-extracts/
http://docs.mongodb.org/manual/reference/program/mongoimport/
http://stackoverflow.com/questions/14150823/python-json-decode-valueerror-extra-data
http://trimc-db.blogspot.com/2015/01/using-python-driver-in-mongodb-for.html
http://api.mongodb.org/python/current/tutorial.html